The claims are rearranged in the ways described below. Regarding referencing non-existing steps D, E, and F, these steps actually exist. When USPTO published my application, the line feeds and carriage returns were removed, and indentations were also removed. It made it hard to read and see these steps being listed. That was why the Examiner thought that these steps did not exist. Actually the last page of the full text image of the publication of my application from USPTO shows correct formatting.

Claim 1 was rearranged in the following way:

1) A period was added at the end of this claim;
2) Added words "in a codeless manner without using of a compiler". These words were in the "Background of Invention" section of the application: "This invention achieves code-less programming by building such a developing and executing environment such that...". The meaning of these words was also implied in the claims 1, 2, 3, and 6.

Claim 2 was rearranged in the following way:

1) A period was added at the end of Claim 2.

Claim 3 was rearranged in the following way:

1) A period was added at the end of Claim 3.
2) Added words "so that the action list becomes an event handler for the event and thus codeless programming is achieved" to explain "event-action-list-mapping". Term "Event-action-list-mapping" is used in my invention in the same meaning of term "event handler" in programming with coding.

Claim 4, 5

1) These two claims were canceled. Their contents were moved into claim 6.

Claim 6 was rearranged in the following way:

1) Claim 4 and claim 5 were canceled and their contents were put in the context building text of claim 6;

2) A period was added at the end of claim 6.

Claim 7 was rearranged in the following way:

1) A period was added after this claim.

Claim 8 was rearranged in the following way:

1) A period was added after this claim.

Claim 9 was rearranged in the following way:

1) A period was added after this claim.

Reply Note 4

Regarding Page 3

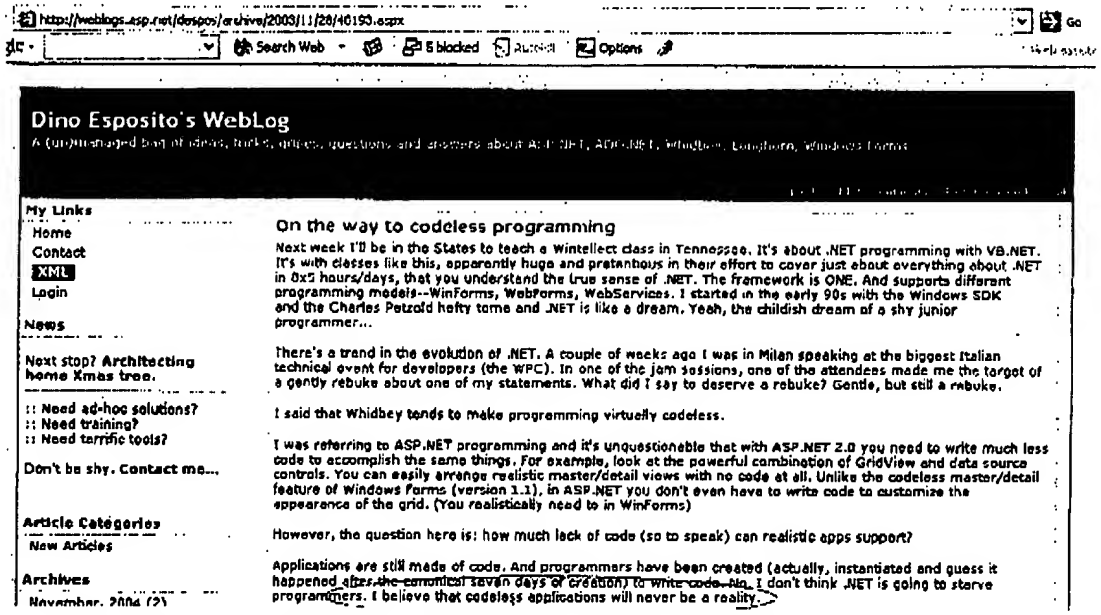
Topic: Claim Rejections

Response Summary:

What taught by Patel, Budd, Linden, and Deitel put all together cannot provide motivations, hints and suggestions leading to what my invention has achieved and claimed: codeless programming without compiling; therefore my invention is not obvious.

Response Details:

1. Patel and Budd did not define the action class and action list class which are precisely defined in my invention; nor did they provide any hints and motivations related to my action class and action list class. My action class and action list class are key components in achieving my codeless programming system which has clear advantages over prior-arts, including taught by Patel and Budd, see Reply Note 6, 7, 8, 9, 10, 12, 13, 14. Therefore my invention is not obvious.
2. Patel and Budd did not use the way described in my invention to do event handling; nor did they provide any hints and motivations related to my way of event handling. My way of event handling is a key component in achieving my codeless programming system which has clear advantages over prior-arts, including taught by Patel and Budd, see Reply Note 10, 15, 16, 17, 18, and 22. Therefore my invention is not obvious.
3. My sorted action list is for execution of actions, one by one, not for action selection. My action class is specifically designed for my way of event handling to achieve codeless programming without compiling. There are no motivations and suggestions in Patel and Budd for creating my action class and action list class; nor did they provide any hints and motivations related to making and executing action class objects and action list class objects. Therefore my invention is not obvious. See Reply Note 7.
4. US Patent 5,850,548 (December 1998) and US Patent 5,862,379 (January 1999) are providing the solutions for the same problem: to create a graphic (preferably codeless) programming system. My invention has clear advantages over these two patents. See **Reply Note 26** below that my invention has clear advantages over US Patent 5,850,548. US Patent 5,862,379 still needs script language while my invention is a complete codeless programming system. "Codeless programming" is still a frontier in software engineering and research. It is not a topic for persons with average skills. Actually in 2003 an expert (Dino Esposito) stated that "I believe that codeless applications will never be a reality." See http://weblogs.asp.net/despos/archive/2003/11/28/40193.aspx.



Clearly codeless programming is much desired in software engineering and is very difficult to achieve. My codeless programming invention has clear advantages over prior-arts and thus my invention could not possibly be obvious to persons with average skills. The software development system using my invention has received much appreciation from users. For example, Albert Mullagulov from Turkmenistan said: "I've just discovered the product you are marketing and, to be honest, I'm amazed at the features of Limnor Application." "Since my programming knowledge (even in VB) is quite restricted, I've been looking for ready-to-use sample codes to design good applications for my business use and when I happened on your application, I realized that it was the very thing I'd sought for.". See http://www.limnor.com/ (go to FAQ) (It is rated 5-stars by several web sites). This product has become the major codeless programming system in the world (searching "codeless programming" in www.google.com, and several search results in the first page will be for this product).

The above points regarding the references will be detailed when responding to each reference below.

Reply Note 5

Regarding Page 4

Topic: Building context for presenting invention in claim 1. Patel pages 265-268, Fig 15.1, Linden page 126.

Response Summary:

Most texts quoted from claim 1 on this page are for building a context to present my invention.

Response Details:

Most texts quoted from claim 1 on this page are for building a context to present my invention. My invention is to define an action class and an action list class as event handler to form a new codeless programming system, which is not obvious for a person with average skills.

Patel pages 265-268 and the figures describe a graphic programming process with Java beans. This

Java bean process cannot be linked to my invention. The following points proved that the approach used by my invention is not obvious:

1. In Patel's process an event of one Java bean is linked directly to a method of another Java bean; the method of the Java bean acts as the event handler. In my invention, an event is linked to an object of action list class (see Claim 3); the members of the action list class are defined by the user interactively (see Claim 1, step A). There is no motivation and suggestion in Patel's teaching to create an action list object as an event handler. Java bean process is not a true codeless process; the development system creates the event linking code; compiling is needed. My process does not involve coding at all, and no compiling is needed.
2. In Patel's process when an event is linked to a method of a Java bean, the development system automatically generates a new adapter class code. Such an adapter class code is unrelated to the action class defined in my invention. My action class is not dynamically created code, and thus no re-compiling is needed for my system to work. My action class can be used for any event handling while Java bean needs different adapter classes for different types of events. Therefore what Patel taught cannot be used or modified to achieve what my invention achieved: codeless programming without compiling.
3. Linden page 126 mentions "every thing is an object". In my approach, an object must have a method to process an action class of my definition, see claim 6, step H3C4: "The said located object class instance carries out the said action method". This is a special requirement only needed for my invention. There are no suggestions and motivations in Linden page 126 to require the generic object to be able to process an action object.

Reply Note 6

Regarding Page 5 – Paragraph 2

Regarding Budd pages 89 – 92

Topic: Action class

Response Summary:

My action and action list classes have deep, wide and fundamental differences from Budd pages 89 – 92. Because of the differences are so fundamental that what taught by Budd cannot be used or modified, directly or indirectly, for the purpose of my invention, and there are not motivations and suggestions from what taught by Budd that can be linked to my invention. Therefore my invention is not obvious.

Response Details:

What currently taught in all literatures including taught by Budd cannot be used or modified for the purpose of my invention, no matter directly, indirectly or as a hint, as shown below:

1. actionListener is an interface. My Action class is a class, not an interface, see Claim 1: "defining an action class and an action list class; the action class has, as its members, action performer, action method, and action data; the action list class contains a sorted list of action

class instances; the action performer is one of the pre-developed object class; the action method is one of the methods supported by the action performer; the action data are the parameters needed by the action method." My action class is a key component in my invention to accomplish codeless programming. It is used to define application behavior and used in handling events (action class itself does not involve events). In Java and Java bean programming, because interface and wrapper class are event type specific it is not possible to modify them to handle all type of events like my action class can; and because interface and wrapper class are used, there is no motivation of come to my idea of action class. In US Patent **5,862,379**, a linking object plus script language are used to achieve codeless (partially codeless because script language is used to define application behavior) graphic programming. Using my action class the application behavior can be controlled without using script language; thus my invention of action class is not obvious; see Reply Note 15. In US Patent ***5,850,548***, property change is used as event linking message to try to make graphical programming system. Limiting program flow behavior to property changes is clearly too restrictive comparing using my action class to define program flow behavior when my action class is forming action list class and used in event handling. See Reply Note 26 for details. This advantage of my invention again proves that my invention of action class is not obvious.

2. In Java if a class implements actionListener it must write code for actionPerformed method, and thus recompiling is needed. In my Action class there is not actual execution code at all. See Claim 1: "the action method is one of the methods supported by the action performer". That is, the actual execution code is defined outside of my action class. Because no coding is needed when defining event handler, this arrangement makes codeless and no-compilation programming possible.
3. In Java if a class implements actionListener then such a class can be passed to a Button object as the Click event handler. But it cannot be used for other kinds of events which expect other interfaces because the Java compiler must match the specific interface. My action class and action list class can be used to handle <u>any</u> events, because in my invention an event is linked not to a function, and thus no need to match event handling interface; this arrangement makes codeless and no-compilation programming possible. See Claim 3: "linking, in response to input from the user, action list instances created in step D to events of the instances of the object classes to form an event-action-list mapping".
4. Because all classes implementing actionListener must be pre-hard-coded, the available actions are limited if used in a codeless graphical manner. My action class does not have execution code itself and it can use <u>any</u> methods supported by all action performer classes which are classes supporting properties-methods-events model; this arrangement makes codeless and no-compilation programming possible.
5. For my invention to work my action class requires an object supporting properties-methods-event model as the action performer, see Claim 1. There is not such a requirement for classes implementing actionListener.
6. In ordinary technologies, executing an event action is simply calling a function, for example, actionPerformed function. In my invention, executing an action can only be done in the

manner defined in Claim 6: "H3c1. Locating the object class instance which is assigned as the action performer for the action; H3c2. Signaling to the said action performer which action method is specified for the action; H3c3. If there are method data specified for the said method of the said located object class instance, the method data are passed to the said object class instance as well; H3C4. The said located object class instance carries out the said action method". This approach does not appear in any literature but it plays a center role in my invention of codeless programming, and thus it cannot be obvious.

7. In ordinary technologies, event parameters, for example, mouse position for a mouse event, are passed into event handler as function parameters directly. In my invention, because event handler is not a function, because event handler is an action list object, event parameters cannot be passed to action list object directly. Claim 7 is my way of using event parameters.

Reply Note 7

Regarding Page 5 – Last paragraph, Page 6 – First paragraph

Regarding Budd pages 227 – 231

Topic: action list class vs menu items

Response Summary:

My action and action list classes are totally unrelated to menu and menu items in Budd pages 227 – 231; my action list class has very narrow and specific definition for executing a set of actions in a specific order, which has nothing to do with menus; therefore my invention is not obvious. What described in Budd pages 227 – 231 cannot be used for building an execution list for a codeless programming system, and thus my invention is not obvious.

Response Details:

Budd is teaching menu items. My action list class has nothing to do with menu items. Menus cannot be used or modified to act as my action list class, and thus my invention is not obvious; as shown below.

1. Menu items are presented to the user to select only one item to execute at runtime. My action list is not a selection list. My action list is an execution list. All actions in the action list will be executed, one by one in a specified order. In this way more than one action can be executed in response to one event. See Claim 6: "sequentially performing each action in the said action list mapped to the said event;". Menu items cannot be used as my action list. So it is clear that menu items are unrelated to my action list.
2. For each menu item a pre-coded-function must be made. In my action list there is not actual execution code. See Claim 1: "the action method is one of the methods supported by the action performer". That is, the actual execution code is defined outside of my action class. That is, menu items have nothing to do with my action list.
3. The ordering of my action list is not for quicker access. It is not for users to visually locate selection, nor to locate an item in the list quickly. I am defining the execution order by specifying which action will be performed first, which action will be performed next, and so on. Execution order is very important in programming. All actions in a list will be performed one by one. See Claim 6. In codeless programming research field no one handles an event

using an action list object before my invention.

4. Menu items are to be selected by the user at run time for just one item. My action list is to be used in such a manner: it is selected at design time when building event-action-list mapping, and it is not for the user to select one action out of the list, it is for the user to select an action list as a whole, at design time, not runtime. See claim 3: "E3. Selecting, in response to input from the user, an action list class instance from the action list class instances created in step 2; E4. Building the mapping relationship between the action-list selected in step E3 and the event selected in step E2;". Therefore menu items and my action list are two unrelated subjects.

Reply Note 8

Regarding Page 6 – The second paragraph

Regarding Budd section 6.2.2.

Topic: Action Performer

Response Summary:

Action Performer concept in my invention has specific meaning not covered by ordinary technologies. My action method and action performer concepts have deep, wide and fundamental differences from Budd section 6.2.2. What taught by Budd cannot be used or modified for the purpose of my invention: codeless programming without compiling. Therefore the innovativeness of my invention is not affected, and my approach is not obvious.

Response Details:

In my invention an action performer class is referring different thing than what ordinary technologies, represented by Budd section 6.2.2., referring to. These fundamental differences made it not possible for what taught by Budd to be used or modified for the purpose of my invention: codeless programming without compiling. The differences are listed below:

1. In my invention an action performer class is a member of action class and it is an instance of pre-developed class which must support properties-methods-events model. See Claim 1. The action performer class must also be able to handle action object, see Claim 6 step H3C4. In Budd section 6.2.2., the pre-developed action performer class is a class which must implement actionListner interface, it does not need to support properties-methods-events model, and thus it does not qualify as an action performer in my invention. Since it is not possible to turn an event type specific actionListner interface into my action class which is event type independent, it is not possible by modifying what taught by Budd to achieve codeless programming my invention achieves, and therefore my approach is not obvious.
2. In my invention an action performer may only perform its own actions, by the nature of my action class. See Claim 1: "the action method is one of the methods supported by the action performer". Therefore the action method can only be for the action performer itself. Because action method is pre-developed for the action performer, it cannot be for handling application specific other objects' events. So the action method has nothing to do with any application specific events. This separation of actions and events make it possible for an action to be used for any events without recimpiling. In Budd section 6.2.2., the code inside actionPerformed

has nothing to do with its "action performer class" – the class implementing this actionPerformed method, FireButtonListener class. The code inside actionPerformed is application specific and for the click event of the Button class, not for FireButtonListener class. The purpose of the code inside actionPerformed is for handling a particular event of a particular object; changing of event handling must involve recompiling. The above fundamental differences in concepts and approaches prove that Budd section 6.2.2 cannot be used or modified to achieve what my invention achieves: codeless programming without compiling. Therefore my invention is not obvious.

3. Actually class FireButtonListener is the way how Java can support functionality of function pointers in C/C++. In C# such functionality is supported through delegations. In my invention, an action performer is like a person who can do certain things. In Budd section 6.2.2., actionPerformed method has nothing to do with its hosting class FireButtonListener but it has every thing to do with an event of another object, the Button object. In my invention, Claim 1 says: "the action performer is one of the pre-developed object class". Since it is pre-developed, its action methods cannot have any thing to do with handling application specific events. Since it is pre-developed and thus has nothing to do with other objects, its action methods can only handle things related to the action performer class itself. That is, in my invention, action methods of an action performer has nothing to do with any events but it has every thing to do with its hosting class, the action performer. We can see that if we try to apply my "action" and "action performer class" concepts to what Budd taught, like the examiner did, the ways of using these concepts are opposite to what my invention is doing. Such opposite ways of using of the concepts proved that it is not possible to modify what Budd taught to do what my invention is doing and achieving: codeless programming without re-compiling. Therefore my invention is not obvious.
4. Action method in my invention is closely related to action performer. So Reply Note 9 applies to this topic entirely.

Reply Note 9

Regarding Page 6 – The third paragraph

Regarding Budd section 6.2.2.

Topic: Action Method

Response Summary:

My action method and action performer concepts have deep, wide and fundamental differences from Budd section 6.2.2, and it is not possible by modifying what Budd taught to achieve what my invention achieves: codeless programming without compiling, therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:

The action method defined in my invention is fundamentally different than what Budd section 6.2.2 presents. There is no obvious link between my invention and what Budd taught. The differences are listed below.

1. actionPerformed in Budd section 6.2.2 is a function with real execution code, and changing the code must involve recompiling. The action method defined in my invention is not a real execution code; it is an indication of one of the methods the action performer supports. See Claim 1: "the action method is one of the methods supported by the action performer", and change of action method does not involve recompiling.
2. actionPerformed in Budd section 6.2.2 once being coded cannot be changed without recompiling. This is determined by the Java compiler and cannot be modified. The action method defined in my invention can be changed at runtime without recompiling the program. See Claim 2: "D1b. selecting a method from the methods supported by the instance of object class selected in step D1a". What Budd taught cannot be used or modified to achieve codeless programming without recompiling, and therefore my invention is not obvious.
3. actionPerformed in Budd section 6.2.2 belongs to an interface. The locating of execution code is done by Java compiler. The action method defined in my invention is located by my system not by a compiler and must be interpreted together with the action performer which is also a member of an action class, see Claim 6 step H3. For example, suppose the 5th method of a Picture object is "Display a picture" and we have an action class with its action performer being a Picture performer and its action method is 5, then this action method indicates the "Display a picture" of the Picture object. Now if its action performer changed to a different object, the action method is still 5 then the action method will represent another method. If the action performer is changed to an object which has fewer than 5 methods then the action method of this action class become invalid. See Claim 1: "the action method is one of the methods supported by the action performer". Because of such fundamental difference of what Budd taught and what my approach achieves, it is not possible to use or modify what Budd taught to achieve what my invention achieves: codeless programming without compiling. Therefore my invention is not obvious.
4. actionPerformed in Budd section 6.2.2 if used as an event handler it is through the interface it belongs to. The event handling is linked by compiler. The action method defined in my invention if used as an event handler it is through the action class it belongs to, not by compiler. See Claim 3: "linking, in response to input from the user, action list instances created in step D to events of the instances of the object classes to form an event-action-list mapping". Note that action list class consists of a sorted list of action class objects. The event handling is done not by compiler, but by a mapping relation the user builds. See claim 6: "H3a. Checking if there is a mapping relationship between an action list class instance and the said event; H3b. If the said mapping relationship exists, sequentially performing each action in the said action list mapped to the said event;". Therefore what Budd taught cannot be used or modified to achieve codeless programming without recompiling. Hence my invention is not obvious.
5. actionPerformed in Budd section 6.2.2 if used as an event handler it can only be used in certain events which expecting the interface it belongs to. The action method defined in my invention if used as an event handler it can be used for any events. See Claim 3, the building of the event-action-list-mapping is the selection of event and action-list; there is no

restrictions on which action lists must be mapped to which events; hence my way of codeless programming is possible.

6. actionPerformed in Budd section 6.2.2 if used as an event handler it is built by developer using compiler. The action method defined in my invention if used as an event handler it is done by the user without compiler. Therefore what Budd taught cannot be used or modified to achieve codeless programming without recompiling. Hence my invention is not obvious.
7. Action method in my invention is closely related to action performer. So Reply Note 8 applies to this topic entirely.

Reply Note 10

Regarding Page 6 – The 4th paragraph

Regarding Budd section 6.2.2.

Topic: Action Data vs ActionEvent

Response Summary:

Action Data is a special way of using data in actions in my invention, which is unrelated to Budd 6.2.2 and thus the innovativeness of my invention is not affected, and my invention is not obvious.

Response Details:

Action Data in my invention is similar to parameters in functions. ActionEvent in Budd 6.2.2 is parameters for events. They are unrelated subjects as shown below:

1. ActionEvent is the data come with the event such as mouse position for a mouse event. Action Data is the data needed to carry out an action. For example, ShowImage action may require an image file name as its action data. See Claim 1: "the action data are the parameters needed by the action method".
2. ActionEvent describes the properties of a specific event. My Action Data has nothing to do with any events. My Action Data are similar to parameters in a function. But such parameters are not for describing event properties. See claim 1: "the action data are the parameters needed by the action method".
3. My Action Data is similar to parameters in a function. But they are not function parameters. My Action Data is a member of my Action class. See claim 1: "the action class has, as its members, action performer, action method, and action data". In this way users may specify action data without compiling, and thus my way of codeless programming is possible.

Reply Note 11

Regarding Page 6 – The 5th paragraph

Page 7 – The 1st paragraph

Topic: Context building text

Response Summary:

Claim 1 includes some context building text which if taking out alone does not form an invention.

Response Details:

Step B and Step C were intended to build a context for claim 1. Other patents use the same way of

presenting claims. Let's use a claim from US Patent ***5,850,548*** to show it:
==Start quote of US Patent ***5,850,548*** claim 1==
What is claimed is:
1. In a computer system, a method for creating an application program using graphical user interface technique, the method comprising:
(a) displaying a form for presenting screen objects on a screen display;
(b) displaying a plurality of prefabricated program components, some of said prefabricated program components corresponding to said screen objects;
==End of quote of US Patent ***5,850,548*** claim 1==

We can see that step (a) and (b) above do not have innovative contents.

Reply Note 12
Regarding Page 7 – The 2nd paragraph
Topic: action list class
Response Summary:
My action and action list classes are totally unrelated to menu and menu items in Budd pages 227 – 231; my action list class has very narrow and specific definition for executing a set of actions in a specific order; therefore the innovativeness of my invention is not affected and my invention is not obvious.
Response Details:
This paragraph is for Claim 2. Claim 2 is for building action list class instance. The rejection of Claim 1 regarding action list is comparing my action list with menu items. "Reply Note 7" describes the differences between my action list and menu items, and proves that menu items are totally unrelated to my action list class. Thus my invention is not obvious.

Reply Note 13
Regarding Page 7 – The 3rd , 4th, and 5th paragraphs
Topic: Action creation vs Patel Figs. 15.1 – 15.4
Response Summary:
My action class is unrelated to Java Bean, therefore the innovativeness of my invention is not affected and my invention is not obvious.
Response Details:
Patel Figs 15.1 – 15.4 are not doing the same function as Step D of my invention is doing. They are unrelated subjects as shown below.

1. Patel Figs 15.1 – 15.4 show the event linking capability of Java Bean. Step D in my invention is the process of creating action class instance and action list instance. My action class and action list class are defined in Claim 1. The definitions of these classes do not involve events and event linking. No one has defined my action class and action list class before the filing of this patent application. Patel Figs 15.1 – 15.4 is totally unrelated to my action class and action list class.

2. Patel Figs 15.1 – 15.4 show event firing Java Bean and event handling Java Been. Step D in my invention is similar to creating a software function in a codeless manner; there is not an event firing object involved in action class and action list class. See Claim 1, the definition of the action class and action list class: "defining an action class and an action list class; the action class has, as its members, action performer, action method, and action data; the action list class contains a sorted list of action class instances;". Action class and action list class by themselves have nothing to do with events.
3. For differences between my way of event linking and Java way of event linking, please see Reply Note 15, Reply Note 16, Reply Note 17, and Reply Note 18.

Reply Note 14

Regarding Page 8 – The second paragraph

Topic: Specify action data vs Budd Fig 13.5

Response Summary:

Budd Fig 13.5 is unrelated with Step D1c in my invention; therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:

Budd Fig 13.5 shows two visual components: TextField and TextArea. Step D1c in my invention is specifying action data. They are unrelated subjects as shown below.

1. TextField and TextArea in Budd Fig 13.5 are visual components. Action Data in my invention are parameters for an action method. Budd Fig 13.5 and Step D1c in my invention are referring to totally different objects;
2. Budd does not show what methods TextField and TextArea have. For example, TextField and TextArea have a method of setText(string). Step D1c in my application requires displaying all methods supported by an object. See Claim 2: "D1b. selecting a method from the methods supported by the instance of object class selected in step D1a; the selected method is used as the action method for the action class instance;"
3. Budd does not show what method parameters are. For example setText(string) method has a string parameter. Step D1c in my application requires showing method parameters. See Claim 2: "D1c. according to the method selected in step D1b, it is known the number and types of the parameters needed for the said method; if one or more parameters are needed for the said method, then one or more dialog-boxes are provided for the user to specify the appropriate parameters for the method". Budd is not talking about codeless programming without compiling and therefore there is no motivation and suggestion to show method parameters to the user.
4. Budd Fig 13.5 simply shows two visual components TextField and TextArea. Users cannot know what the inputs are or what inputs are supported. Actually what are the inputs are not defined in Budd at all. Step D1b in my invention is to show all methods supported and Step D1c is to show all parameters for the selected method.

Reply Note 15

Regarding Page 8 – The third paragraph
Topic: Event linking vs Patel Fig 15.7
Response Summary:
Event-linking in my invention has fundamental differences than ordinary technologies represented by Patel Fig 15.7 and other inventions such US Patent 5,862,379 and SoftWire (http://www.softwire.com/). Therefore the innovativeness of my invention is not affected and my invention is not obvious.
Response Details:
The differences are listed below.

1. Patel Fig 15.7 (including Fig 15.1 to 15.5 and Java bean programming), U.S. Patent 5,862,379 and SoftWire are similar in that all of them take the major programming task as to link the event firing object (source object) to the event consuming object (destination object). They adopt such a programming pattern so that it is possible to visually build program flow path. There should be people like this way of programming. But the limitation of this way of programming is the lack of flexibility and thus limits the power of the programming, because no parameters can be used when calling a method. For example if a class has a method of moving itself on the screen to point (x, y) then there is no way it can be used by such programming approach because we cannot pass parameters (x, y). U.S. Patent 5,862,379 uses a script language to try to provide more flexibility. In Java way of doing it (taught by Patel, Budd as referenced) in addition to the above limitation, recompiling is also needed. My invention uses a totally different approach and does not involve the concepts of "source object" and "destination object", and does not use script languages. As described in my application, my invention puts an action performer concept in the center of a programming task; all software tasks to be programmed into an application belong to one or more action performers; any computer action must be carried out by an action performer; an action performer just carries out its own actions. My invention treats an event as a trigger to signal a set of action performers to start carrying out their own actions one by one. This reflects a fact that when a performer receives a signal to start performing an action, in some cases it does not matter much where the signal is come from. Using the programming tool my invention presents, when a user wants to make a software application to do some things, the user asks "who can do these things?" The answer is to find the right Action Performers provided by the programming tool to do the things. So the user selects and creates the action performers needed for her/his application. The user then asks "when each thing should be done?" The answer is to pick the right events. Programming process is thus simple and straight forward. Therefore this invention's approach will be much more intuitive to a wide range of non-technical oriented users. Because my action class has action data member for action method parameters my approach does not have the limitation of prior-arts. Without my action class my way of codeless programming cannot be achieved, thus the prior-arts cannot be adjusted to achieve the codeless programming my invention achieved.
2. In Patel Fig 15.7 (including Fig 15.1 to 15.5 and Java bean programming) when an event firing object is linked to an event handling object, a new class is automatically created to

make the link. The new class will be compiled. In my invention an event is linked to an action list object, not to an event handling object; there is no need to create new code and new recompiling. See Claim 3: "linking, in response to input from the user, action list instances created in step D to events of the instances of the object classes to form an event-action-list mapping;" and Claim 6: "H3. Responding to each event fired by each object class instance; wherein step H3 comprises: H3a. Checking if there is a mapping relationship between an action list class instance and the said event; H3b. If the said mapping relationship exists, sequentially performing each action in the said action list mapped to the said event; H3c. Each action in the said action list is performed by the following steps: H3c1. Locating the object class instance which is assigned as the action performer for the action; H3c2. Signaling to the said action performer which action method is specified for the action; H3c3. If there are method data specified for the said method of the said located object class instance, the method data are passed to the said object class instance as well; H3C4. The said located object class instance carries out the said action method." So in my invention, no need new code and re-compiling when linking action list objects to events and when executing the action list object. This big advantage over prior-arts proves that my approach is not obvious.

3. In Patel Fig 15.7 (including Fig 15.1 to 15.5 and Java bean programming) when an event firing object is linked graphically to an event handling object, no parameters can be used in the event handler. For example if a ShowImage method needs a file path as the parameter then such a method cannot be graphically linked to events. In my invention because what linked is an action list object, parameters can be passed. See Claim 6: "...H3c3. If there are method data specified for the said method of the said located object class instance, the method data are passed to the said object class instance as well;". This big advantage over prior-arts proves that my approach is not obvious.
4. In Patel Fig 15.7 (including Fig 15.1 to 15.5 and Java bean programming) when an event firing object is linked to an event handling object, an event can only be linked to one event handler object. In my invention because an event is linked to an action list object, many event handler objects can be involved. See Claim 3 for linking to action list object and Claim 1 for action list class and action class definitions.
5. Like Patel Fig 15.7, U.S. Patent 5,862,379 also links event firing object to event handling object, but added a script language for adding more execution behaviors. The script language is a commercially available product in the market. What patented in U.S. Patent 5,862,379 was the idea of adding script language when linking event firing object to event handling object. U.S. Patent 5,862,379 has much more closeness to ordinary technologies taught by Patel and Budd. Like US Patent 5,862,379, my invention also proposed a new idea of doing event linking. But my invention from the very beginning is not linking an event firing object to an event handling object. My invention invented a new action class and action list class for the event linking. My invention has clear advantage of truly codeless (no need script language) and therefore the innovativeness of my invention is higher than US Patent 5,862,379, and thus it cannot be obvious.

Reply Note 16

Regarding Page 8 – The 4th paragraph, Page 9 – The first paragraph

Topic: Event linking vs Patel Fig 15.15

Response Summary:

Patel Fig 15.15 (and the Java bean programming) cannot be modified to match the event handling scheme my invention precisely defines to gain the advantages of my scheme; therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:

1. Patel Fig 15.15 shows linking between events and event handlers by linking event firing objects with event handling objects. One event is linked to one method of an object as an event handler. In my invention because action list class is used as event handler, more than one object can be involved when handling an event. See Claim 3 for linking to action list object and Claim 1 for action list class and action class definitions.
2. In Patel Fig 15.15 showing linking between events and event handlers, parameters cannot be used when calling event handler function. Here I am not talking about event parameters; I am talking about action data similar to function-parameters. For example a ShowImage action may need an image file path as action data. My action class includes action data. See Claim 1 about my action class definition, and Claim 3 the linking of event to action list class. This big advantage of my invention proves that my invention is not obvious.
3. For differences between my way of event linking and Java way of event linking, please see Reply Note 15, Reply Note 17, and Reply Note 18.
4. Step E1 and E2 describe event selection for doing event-action-list mapping. The whole claim includes all steps from E1 to E4. Step E1 and E2 themselves do not form the claim. U.S. Patent 5,862,379 uses the same way to make a claim. See below:

U.S. Patent 5,862,379 Claim 1:

A method for creating an application program, the method implemented on a computer system having a display screen and an input device, the input device controllable by a user to create visual representations on the display screen, the method/comprising:

A. defining and supporting a set of object classes, the set of object classes including a linking object class;

B. selecting, in response to input from the user, a first one of the object classes;

C. generating, in response to the user drawing a first visual representation on the display screen, a source object, the source object being an instance of the first object class and having a first set of associated default events and property settings;

D. selecting, in response to input from the user, a second one of the object classes;

E. generating, in response to the user drawing a second visual representation on the display

screen, a destination object, the destination object being an instance of the second object class and having a second set of associated default events and property settings;

F. selecting, in response to input from the user, the linking object class;

G generating, in response to the user drawing a third visual representation connecting the first and second visual representations, a linking object, the linking object being an instance of the linking object class and having a set of user selectable predefined behaviors used by the destination object, the predefined behaviors being responsive to at least one of the first set of associated events; and

H. displaying, in response to input from the user, the default events and property settings, of selected of the source, destination, and linking objects.

In the above U.S. Patent 5,862,379 Claim 1, steps B, C, D, and E do not have innovative contents. But they are necessary to form this claim.

Reply Note 17

Regarding Page 9 – The second paragraph

Topic: Event linking vs Patel Fig 15.9 in view of Fig 15.6

Response Summary:

Patel Fig 15.9 and Fig 15.6 are unrelated to the event linking scheme my invention precisely defines; therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:

1. Patel Fig 15.9 shows a tool box showing all Java Beans. This fig has nothing to do with event linking. Claims in my invention are not related to this fig. Therefore Patel Fig 15.9 does not demerit my claim step E3.
2. Patel Fig 15.6 shows event selection only. It corresponds to step E1 and E2 in my claim 3 only but is unrelated to step E3 and E4. My claim 3 includes step E1, E2, E3, and E4. As quoted for this paragraph (step E3), my invention defines linking an event to an action list class. Action class and action list class are my inventions. Therefore Patel Fig 15.6 is unrelated to my claim step E3. Without my action list class and linking event to it, the event linking when done in a codeless/graphical manner, as taught by Patel and Budd, can only be linked to a method without using its parameters. With such a sever limitation complete codeless programming is not possible.
3. More comments on my event linking and Java programming can be found in Reply Note 15, Reply Note 16 and Reply Note 18.

Reply Note 18

Regarding Page 9 – The third paragraph

Topic: Event linking vs Patel Fig 15.6 and view of Fig 15.7

Response Summary:

What Patel taught, including Fig 15.6 and Fig 15.7, cannot be used or adjusted to match the event linking scheme defined precisely in my invention; and cannot take the advantages of my scheme; therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:

1. Patel Fig 15.6 shows event selection only. It corresponds to step E1 and E2 in my claim 3 only. My claim 3 includes step E1, E2, E3, and E4. As quoted for this paragraph (step E4), my invention defines linking an event to an action list class. Action class and action list class are my inventions. Therefore Patel Fig 15.6 is unrelated to my claim step E3 and E4. Without my action list class and linking event to it, the event linking when done in a codeless/graphical manner, as taught by Patel and Budd, can only be linked to a method without using its parameters. With such a sever limitation complete codeless programming is not possible.
2. Patel Fig 15.7 shows linking between events and event handlers by linking event firing objects with event handling objects. One event is linked to one method of an object as an event handler. In my invention because action list class is used as event handler, more than one object can be involved when handling an event. See Claim 3 for linking to action list object and Claim 1 for action list class and action class definitions. Therefore Patel Fig 15.7 does not link to my claim step E4. **Reply Note 15** provides detailed descriptions on how Patel Fig 15.7 does not link to my claim step E.
3. More comments on my event linking and Java programming can be found in Reply Note 16 and Reply Note 17.

Reply Note 19

Regarding Page 9 – The 4th paragraph

Topic: Object selection vs Patel Fig 15.6

Response:

Claim 4 and Claim 5 were needed for building a context for claim 6. I'll cancel claim 4 and 5, and merge the contents of claim 4 and claim 5 into claim 6. All claim text is not changed.

Reply Note 20

Regarding Page 9 – The 5th paragraph, Page 10 – The first paragraph

Topic: Claim is so unclear

Response:

Claim 4 and Claim 5 were needed for building a context for claim 6. I'll cancel claim 4 and 5, and merge the contents of claim 4 and claim 5 into claim 6. All claim text is not changed.

Reply Note 21

Regarding Page 10 – The second paragraph

Topic: Claim is so unclear

Response Summary:

The claim 4, 5, and 6 are merged into one claim.

Response Details:
Claim 4 and Claim 5 were needed for building a context for claim 6. I'll cancel claim 4 and 5, and merge the contents of claim 4 and claim 5 into claim 6.
All claim text is not changed.

Reply Note 22
Regarding Page 11 – The second paragraph
Topic: Event data buffer
Response:
Without Claim 7 My codeless programming invention would have had a big limitation comparing to ordinary technologies with computer coding like C/C++, Visual Basic, Pascal, or C#, etc. In ordinary technology when doing coding, an event handler is a function and the function parameters are event data, for example, mouse position for a mouse event. In those event handlers, event data are available to event handling code inside event handling function. But in my invention, my event handler is not a function. My event handler is an action list class object. Unlike ordinary technologies with coding and compiling, my action list objects are not compiled with events and therefore the event data are not available at compile time. To make the event data available to all actions in an action list object, my invention uses a data buffer to store the event data. It is like a global variable available to all actions. When the user is creating an action, the user may select the event data as the action data. That is my unique way to use event data in my invention.

Reply Note 23
Regarding Page 12 – The first paragraph, Deitel page 418-420
Topic: dynamic binding vs dynamic event linking
Response Summary:

Polymorphism is unrelated to event handling and cannot be used to do runtime event handler switching; therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:
Claim 8 in my invention is related to dynamic event handler switching. Deitel page 418-420 is related to polymorphism in Object Oriented Programming. Claim 8 and polymorphism are two unrelated topics, as shown below.

1. Polymorphism is for same function name to refer to different pieces of execution code at runtime; it has nothing to do with event handling; it cannot be used for dynamically changing event handling function. Claim 8 in my invention is to build a method to change the event-action-list mapping at runtime; it has nothing to do with polymorphism. Polymorphism is unrelated to Claim 8.
2. Later-binding used in Polymorphism is to dynamically determine which version of the function to call based on the object type. For example if a pointer p is referencing a class instance and compiled with code p.Function1(); at runtime, p may reference different objects of the subclasses of the compiled class. The subclasses may override Function1(), and the

overridden versions of Function1() will be called. Claim 8 in my invention is an action method with 2 parameters. Parameter 1 identifies the event. For example, if a button supports 10 events and the first event is Mouse Move, then 0 could be used to identify Mouse Move event for the button. Parameter 2 identifies an action list object. Executing this action will build an event-action-list map between the event identified by the first parameter and the action-list identified by the second parameter. On executing this action, the action-list becomes the event handler for the event. There is no class derivation and function overriding involved in my invention and thus my claim has nothing to do with Polymorphism and later-binding. There is no event handling involved in Later-binding and Polymorphism and thus later-binding and Polymorphism has nothing to do with my invention.

3. To better describe the feature defined by claim 8 of my invention, let me use an example. Assumptions: Suppose we have an action list A which has an action to play a sound, and an action list B which has an action to show an image. Suppose we have a Button object which supports a Click event. Suppose at design time we build an event-action-list mapping between the Click event of the button and action list A. Now at runtime clicking the button will play a sound. Now let's see how the method in this claim works. Suppose the Button object supports the method defined in this claim. At design time, we may create an action object X using this method and set the first parameter to be the Click event, and set the second parameter to be action list B. Now at runtime, if action X is executed (we disregard when to execute action X), clicking the button will not play a sound because action list A is no longer the event handler for Click event of this button; clicking the button will show an image because action list B is now the event handler. That is, before executing action X, clicking the button will play a sound; after executing action X, clicking the button will show an image.

Reply Note 24

Regarding Page 12 – The second paragraph

Topic: claim unclear

Response Summary:

An example is provided to explain this claim.

Response Details:

The steps in this claim defines the way the method defined in claim 8 is executed. Using the example in Reply Note 23, item 3, action object X is created using the method defined in claim 8. The steps in this claim define how such an action (action X) is executed. Let me explain it below.

K. At the runtime, when an object class instance is asked to perform the said method, the said object class instance uses the first parameter of the said method to locate its event;

In the step K, "object class instance" in the example of action object X refers to the Button object because the Button is the action performer of action object X. Because the first parameter of action object X refers to the Click event of the Button, "uses the first parameter of the said method to locate its event" actually locates the Click event of the Button.

L. The said object class instance uses the second parameter of the said method to locate the action list;

Because the second parameter of action object X refers to action list B, "uses the second parameter of

the said method to locate the action list" actually locates action list B.
M. The said object class instance rebuilds the event-action-list map using its event located in step K and the action list located in step L.
In the case of action X, the event-action-list map built will be between Click event and action list B, and hence action list B now becomes the event handler of the Click event of the Button. Clicking the Button will cause action list B to be executed.

Reply Note 25
Regarding Notice of References Cited – US-5,710,894
Topic: US Patent – 5,710,894
Response Summary:
US Patent 5,710,894 is not related to my invention; and the innovativeness of my invention is not affected and my invention is not obvious.
Response Details:
US Patent 5,710,894 is for a feature used in computer simulators. My invention is for a generic-purpose codeless graphic application programming.
US Patent 5,710,894 and my invention have only one thing in common: both systems do not use a compiler and do not use computer languages. Apart from this common thing, nothing is in common between both systems.
What US Patent 5,710,894 patented was an idea of grouping classes and objects. It is to use a class as a container (called a "jar"), and use this container to include other classes and objects, and also exclude other objects and classes.
US Patent 5,710,894 also proposes (claims) a graphic user interface to manage the "jar": creating the "jar"; adding/deleting items in the jar.
US Patent 5,710,894 also proposes (claims) to apply "rules" to jar and apply the "rules" to all objects/classes in the jar.
US Patent 5,710,894 patented the above ideas. No implementation details are given. There is no mentioning of what are "rules" and how to apply and use "rules". There is no mentioning of how graphic user interface is implemented to manage the jars.
The above claims have nothing to do with my invention. My invention has nothing to do with these claims.
US Patent 5,710,894 is for the purpose of collectively applying "simulation rules" to a group of objects, and the grouping of the objects can be controlled by users. This idea has nothing to do with doing generic-purpose codeless graphic programming, which is what my invention is for.
My invention precisely defines action and action list classes. US Patent 5,710,894 does not need and does not have action and action list ideas.
My invention uses an action list as event handler. US Patent 5,710,894 has nothing to do with events and event handling.

Reply Note 26
Regarding Notice of References Cited – US-5,850,548

Topic: US Patent – 5,850,548

Response Summary:

The differences between US Patent 5,850,548 and my invention are deep, wide and fundamental; therefore the innovativeness of my invention is not affected and my invention is not obvious.

Response Details:

The differences between US Patent 5,850,548 and my invention are deep, wide, and fundamental, and are listed below.

1. US Patent 5,850,548 creates programs by linking properties of objects. One property change is passed to a linked property and forms program flow. Limiting program flow to property changes is a clear drawback of lacking flexibility. In my invention program flow is not just initiated by property changes. In my invention program flow is initiated by event firing, and events are linked to action list object. Action class and action list class have concrete definitions. An event firing caused the linked action list to be executed; this is obviously a much more powerful and flexible way of programming. Hence my invention must not be obvious.
2. US Patent 5,850,548 uses only primitive data as messages passing between properties when a property changes. It is not specified how each linked property should respond to messages sent to it. The claims of US Patent 5,850,548 do not say how user can change how the property should respond to messages. In my invention, the user may create event-action-list mapping and thus define how to respond to each event. See Claim 3: "linking, in response to input from the user, action list instances created in step D to events of the instances of the object classes to form an event-action-list mapping". In my invention data passing between objects are not limited to primitive data. See Claim 1: "the action data are the parameters needed by the action method"; it implies that my action data can be of any types because method parameters of cause can be of any data types. These advantages of my invention over US Patent 5,850,548 prove that my invention is not obvious.
3. Another major idea presented by US Patent 5,850,548 is to create nested component. This idea plays no role in my invention.

Reply Note 27

Regarding Notice of References Cited – Patel et al. p.263-278

Topic: Java Bean and Java programming

Response Summary:

My invention has deep, wide, and fundamental differences from what Patel taught and thus my invention is not obvious.

Response Details:

See Reply Note 13, Reply Note 15, Reply Note 16, Reply Note 17, Reply Note 18, Reply Note 19.

Reply Note 28

Regarding Notice of References Cited – Budd.

Topic: Java event model and Java programming
Response Summary:

My invention has deep, wide, and fundamental differences from what Budd taught and thus my invention is not obvious.

Response Details:

See Reply Note 6, Reply Note 7, Reply Note 8, Reply Note 9, Reply Note 10, Reply Note 12, Reply Note 14.

Reply Note 29

Regarding Notice of References Cited – Deitel et al. p.418-420.
Topic: Polymorphism and Java programming
Response Summary:

My invention has deep, wide, and fundamental differences from what Deitel taught and thus my invention is not obvious.

Response Details:

See Reply Note 23.

Reply Notes 30

Regarding Notice of References Cited – Linden pp.126-128.
Topic: Generic object and Java programming
Response Details:

"pre-developed object classes" and "one generic class" mentioned in my claim 1 and mentioned in Linden page 126 are indeed the same concepts. But my object class must have the capability of processing action object, see Claim 6 step H3C4: "The said located object class instance carries out the said action method." The action class is defined in my claim 1: "defining an action class and an action list class; the action class has, as its members, action performer, action method, and action data; the action list class contains a sorted list of action class instances; the action performer is one of the pre-developed object class; the action method is one of the methods supported by the action performer; the action data are the parameters needed by the action method." Linden pages 126-128 have no contents related to my invention. There are no motivations and suggestions in Linden pages 126-128 for creating codeless programming system. Therefore my invention is not obvious.

Claims

1. A method for creating an application program in a codeless manner without using a compiler, the method implemented on a computer system having persistent storage, a display screen and one or more input devices, the input devices controllable by a user to create visual representations on the display screen, the method comprising following steps:

A. defining and supporting a set of pre-developed object classes, the said pre-developed object classes are all derived from one generic class which supports a property-method-event model; defining an action class and an action list class; the action class has, as its members, action performer, action method, and action data; the action list class contains a sorted list of action class instances; the action performer is one of the pre-developed object class; the action method is one of the methods supported by the action performer; the action data are the parameters needed by the action method;

B. generating and graphically displaying, in response to input from the user, instances of the object classes from the said pre-developed object classes;

C. setting, in response to input from the user, each property of each instance of the object classes created in step B.

2. The method of claim 1 further comprising the step of:

D. creating, in response to input from the user, instances of the action list class which contains a sorted list of instances of the action class; wherein step D comprises following steps

D1. Creating, in response to input from the user, each action class instance of each action list class instance; and wherein step D1 comprises:

D1a. Selecting an instance of object class from the instances of object classes created in step B; the said selected instance of object class is used as the action performer member for the action class instance;

D1b. selecting a method from the methods supported by the instance of object class selected in step D1a; the selected method is used as the action method for the action class instance;

D1c. according to the method selected in step D1b, it is known the number and types of the parameters needed for the said method; if one or more parameters are needed for the said method, then one or more dialog-boxes are provided for the user to specify the appropriate parameters for the method.

3. The method of claim 1 further comprising the step of

E. linking, in response to input from the user, action list instances created in step D to events of the instances of the object classes to form an event-action-list mapping so that the action list becomes an event handler for the event and thus codeless programming is achieved; wherein step E comprises:

E1. Selecting, in response to input from the user, an instance of object class from the existing instances of object classes;

E2. Selecting, in response to input from the user, an event from the events supported by the object class instance selected in step E1;

E3. Selecting, in response to input from the user, an action list class instance from the action list class instances created in step 2;

E4. Building the mapping relationship between the action-list selected in step E3 and the event selected in step E2.

~~4. The method of claim 1 further comprising the step of:~~
~~F. Selecting, in response to input from the user, a set of object class instances to be specified as the "initially active object class instances" usually the object class instances presented on the first application screen presentation is such a set of the "initially active object class instances".~~

~~5. The method of claim 1 further comprising the step G. saving to the computer persistent storage the object class instances created in steps A, B and C, the action list class instances created in step D, the mapping relationship built in step E between the events of object class instances and the action lists, indication of which object instances are the "initially active object class instances" as specified in step F;~~

6. The method of claim 1 further comprising the step of:

F. Selecting, in response to input from the user, a set of object class instances to be specified as the "initially active object class instances" usually the object class instances presented on the first application screen presentation is such a set of the "initially active object class instances";

G saving to the computer persistent storage the object class instances created in steps A, B and C, the action list class instances created in step D, the mapping relationship built in step E between the events of object class instances and the action lists, indication of which object instances are the "initially active object class instances" as specified in step F;

H. an execution environment; wherein step H comprises:

H1. Reading back from the computer persistent storage the object class instances created in steps A, B and C, the action lists created in step D, the mapping relationship built in step E between the events of the object class instances and the action list, indication of which object instances are the "initially active object class instances" as specified in step F;

H2. Creating and displaying the said "initially active object class instances";

H3. Responding to each event fired by each object class instance; wherein step H3 comprises:

H3a. Checking if there is a mapping relationship between an action list class instance

and the said event;

H3b. If the said mapping relationship exists, sequentially performing each action in the said action list mapped to the said event;

H3c. Each action in the said action list is performed by the following steps:

H3c1. Locating the object class instance which is assigned as the action performer for the action;

H3c2. Signaling to the said action performer which action method is specified for the action;

H3c3. If there are method data specified for the said method of the said located object class instance, the method data are passed to the said object class instance as well;

H3C4. The said located object class instance carries out the said action method.

7. The method of claim 1 further comprising the step of:

I. A context-data buffer which saves event parameter data such as mouse position in mouse movement events; every time an event is fired, before an action list is executed as an event handler, the said context-data buffer is filled with the said event parameter data;

J. The said context-data buffer is available for the user to pick as the method data in step D1c.

8. A method designed for object classes to dynamically change their event-action-list mapping at the runtime; any object classes may choose to support or not to support the said method; the said method has two parameters; the first parameter is the event identifier which identifies an event supported by the object class which is the owner of the said method; the second parameter is the action list class instance identifier which identifies an action list class instance.

9. The method of above claim further comprising the step of:

K. At the runtime, when an object class instance is asked to perform the said method, the said object class instance uses the first parameter of the said method to locate its event;

L. The said object class instance uses the second parameter of the said method to locate the action list;

M. The said object class instance rebuilds the event-action-list map using its event located in step K and the action list located in step L.